Data Classification using Carbon-Nanotubes and Evolutionary Algorithms

E. Vissol-Gaudin, A. Kotsialos, M. K. Massey, D. A. Zeze,
C. Pearson, C. Groves and M. C. Petty
E-mail:{*eleonore.vissol-gaudin, apostolos.kotsialos, m.k.massey, d.a.zeze, christopher.pearson, chris.groves, m.c.petty*}*@durham.ac.uk*

School of Engineering and Computing Sciences, Durham University
Stockton Road, Durham DH1 3LE, United Kingdom

Abstract. The potential of Evolution in Materio (EiM) for machine learning problems is explored here. This technique makes use of evolutionary algorithms (EAs) to influence the processing abilities of an unconfigured physically rich medium, via exploitation of its physical properties. The EiM results reported are obtained using particle swarm optimisation (PSO) and differential evolution (DE) to exploit the complex voltage/current relationship of a mixture of single walled carbon nanotubes (SWCNTs) and liquid crystals (LCs). The computational problem considered is simple binary data classification. Results presented are consistent and reproducible. The evolutionary process based on EAs has the capacity to evolve the material to a state where data classification can be performed. Finally, it appears that through the use of smooth signal inputs, PSO produces classifiers out of the SWCNT/LC substrate which generalise better than those evolved with DE.

1 Introduction and Background

Evolution-in-materio (EiM) is an Unconventional Computing (UC) technique which focuses on exploiting the underlying properties of materials to bring them to a computation inducing state [12]. Contrary to traditional computing with Metal-Oxide-Silicon-Field-Effect-Transistor (MOFSET) technology, where everything is designed, produced and programmed very carefully, EiM uses a bottom up approach where computation is performed by the material without having explicit knowledge of its internal properties [13].

The idea of EiM can be found in early work of G. Pask [2] which was concerned with growing an electrochemical ear. More recent work [21], is based on observations made when evolutionary algorithms (EAs) were used for designing electrical circuits on Field-Programmable-Gate-Arrays (FPGAs). The resulting circuit topologies were influenced by the material of the board used. Because of feedback provided by the iterative nature of stochastic optimisation interacting with the material, the identified solutions were based on the specific FPGA's properties that were unaccounted for during the board's design. EiM replaces

the FPGAs with un-configured material systems favouring exploitation of some physical property by a search algorithm [12].

Here, using an iterative process, the material is configured until it reaches a state where a pre-specified scheme of interaction is uniquely translated as a computational input/output relationship. Viewing this iterative process as material training, this type of EiM requires the selection of finite training and verification datasets. Since the problem is about a computation, the datasets consist of known input/output pairs from its domain of definition and range, respectively. The training process requires the repetitive application of computation inputs sent to the material and measurements of its corresponding response. Measured responses are translated into computation outputs, which allows the definition of an error function. The physical property measured and the interpretation scheme of the material's response used for translating it into a computation output are pre-specified and fully known before the training process starts.

There are two types of incident signals on the material. Computation inputs, which are used to represent the arguments of a computation, and configuration inputs, which are used for changing the material's properties. Modulation of the incident signals is controlled by an error minimising optimisation algorithm, which explores the problem's search space. The search space itself, is a hybrid of the material's physical state and the subspace spanned by the independent configuration inputs. Hence, the optimisation algorithm aims at configuring the material at a particular state by finding the optimal configuration inputs producing that material state, the response of which can be uniquely translated into a computation. In effect, EiM is a bottom up approach for producing a computing device where the exact architecture, or material state, remains unknown. Reservoir computing is based on similar notions [5, 10].

EiM has a broad scope and can be divided in four inter-dependant dimensions: (a) the type of material used, (b) the physical property manipulated to obtain a computation, (c) the computational problem itself and (d) the optimisation algorithm used for solving the corresponding problem. Figure 1 illustrates the basic concept.

An algorithm selects a set of configuration inputs. Computation inputs from the training dataset are sent to the material; its response is recorded for each input and is translated into a computation output. For each input/output pair, an error is calculated to allow an objective function evaluation. This objective function is minimised by a derivative-free optimisation algorithm.

In our implementation, the evolvable material is connected to a computer via an *mbed* micro-controller fixed on a custom-made motherboard. Configuration and computation input signals are constant voltage charges applied by the *mbed* to the material and outputs are direct current measurements. Voltages are sent to the material through a set of Digital-to-Analogue-Converters (DACs) fixed on the motherboard. They are connected to the array of gold micro-electrodes shown in Figure 1 deposited on a glass slide using etch-back photolithography. The material blend is drop-deposited within a nylon washer (2.5 mm internal diameter) fixed to this platform for material/electronics interaction.

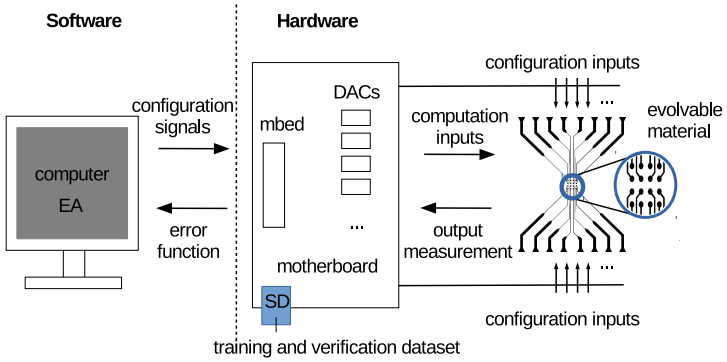


Fig. 1: EiM concept and electrode array ($50\mu m$ contacts, $100\mu m$ pitch)

Different organic and inorganic media have been used as materials, such as slime moulds [7], bacterial consortia [1], cells (neurons) [18], liquid crystals (LC) panels [6] and nano-particles [3]. Single walled carbon nanotubes (SWCNT) based materials have shown the potential to solve computational problems [8, 11, 22, 14, 15]. In [19] it is argued that inorganic materials make a better medium for unconventional computing exploration. Following this argument, as well as results in [22], a mixture of SWCNT and LC in liquid form is used here.

These types of materials have a very complex structure and the development of analytical or stochastic models of their behaviour is very difficult. In their absence, EiM treats them as black boxes, leading to the use of derivative free population based stochastic search algorithms for solving the training problem. Here, a particle swarm optimisation (PSO) [9] and an implementation of differential evolution (DE) [17] are used, which will be referred to as EAs.

Several candidate computational problems can be used in the context of EiM. A more comprehensive review of potential problems can be found in [16]. The problem considered here is a simple binary data classification with different degrees of separation and data distributions.

2 Evolved Material

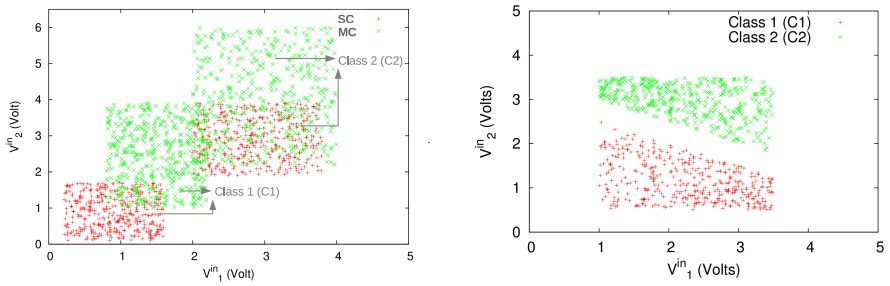
A mixture of SWCNT and LC, where nanotubes are dispersed in liquid crystals at varying concentrations, is used. SWCNT are both semiconducting and conducting; the samples used contain less than 15% impurities (according to vendor specifications) as residual from the catalytic growth process.

It is shown in [22] that SWCNTs tend to bundle under an applied electric field, establishing a percolation path between electrodes. The greater length of these bundles or "ropes" with respect to the dimensions of LC molecules suggests that they are not highly influenced by movement of the latter. The purpose of a LC matrix is therefore to provide a fluid medium in which the SWCNTs can

move in response to the field. Formation of percolation paths is variable and reconfigurable allowing the creation of complex electrical networks. This adds an extra dimension to the problem, compared to previous experiments where SWCNTs were mixed with a solid polymer [11, 8].

3 The Classification Problem

Three variants of a binary data classification problem are considered based on different 2-dimensional datasets. A typical training and verification procedure is followed and the corresponding datasets have $K_t = 800$ and $K_v = 4000$ members. Figure 2 (a) shows the training datasets for the separable (SC) and merged (MC) classes and (b) for the V1 class (V1C). The units of the two computation inputs are in Volts. When a particular pair is used, the two electrodes reserved to receive computation inputs are charged with the corresponding voltages. SC and MC data are organised into two different squares; the SC ones are not overlapping and are placed at a distance, whereas the MC ones overlap slightly. V1C's data are completely separable, but they are arranged diagonally so as to increase the problem's difficulty. After training using those datasets, the material must be in such a state so as to infer the class (C_1 or C_2) for any input pair randomly selected from the verification dataset. Effectively the objective is to evolve an analogue machine, capable of distinguishing the class an input belongs to.



(a) (b)

Fig. 2: (a) SC and MC and (b) V1C training datasets.

4 Problem Formulation

Evolution of such a device is formulated as an optimisation problem. There are sixteen connections on the micro-electrode array twelve of which are used. Two of those are used for sending computation inputs as voltage pulses of amplitude $\mathbf{V}^{in} = (V_1^{in}, V_2^{in})$ and eight are used for sending configuration voltages as pulses

within the range $V_j \in [V_{\min}, V_{\max}]$, $j = 1, \ldots, 8$. The remaining two connections are reserved for measuring outputs currents $\mathbf{I} = (I_1, I_2)$ (A) when the material has been sent $\mathbf{V}^{in}$ and is under charge of the V_j's.

By considering as a decision variable only the possible locations where the two components of $\mathbf{V}^{in}$ are applied and using a simple increasing index scheme for assigning configuration voltages (e.g. if V_1^{in} is assigned to electrode 3 and V_2^{in} is assigned to 5, then the following assignment for the configuration inputs takes place: $V_1 \to 1$, $V_2 \to 2$, $V_3 \to 4$ $V_4 \to 6$, $V_5 \to 7$, $V_6 \to 8$, $V_7 \to 9$ $V_8 \to 10$) then there are ${}^{10}P_2 = 90$ possible connection assignments. A continuous variable $p \in [1, 90]$ is defined and updated by the EA used rounded to the nearest integer during the iterations.

The optimisation problem's vector of decision variables is defined as

$$\mathbf{x} = [V_1 \ldots V_8 \ R \ p]^T \tag{1}$$

where R is a scaling factor. It is for a specific electrode assignment p and set of configuration voltages V_j, that the material's response to an input $\mathbf{V}^{in}$ is recorded. The response is a pair of measurements $\mathbf{I} = (I_1, I_2)$ (A) of the direct current at the two output locations, which are the basis of a comparison scheme using R for deciding the class $\mathbf{V}^{in}$ belongs to.

Let $\mathbf{I}^{(k)}$ denote the pair of direct current measurements taken when input data $\mathbf{V}^{in}(k)$ from class C_i, $i = 1$ or $i = 2$, are applied *while* the material is subjected to configuration voltages $V_j^{(k)}$. $\mathbf{V}^{in}(k)$ and $V_j^{(k)}$ are applied according to electrode assignment number $p^{(k)}$ and scaling factor $R^{(k)}$ is used. Also, let $C(\mathbf{V}^{in}(k))$ denote $\mathbf{V}^{in}(k)$'s real class and $C_M(\mathbf{V}^{in}(k), \mathbf{x})$ the material's assessment of it calculated according to the following rule:

$$C_M(\mathbf{V}^{in}(k), \mathbf{x}) = \begin{cases} C_1 \text{ if } I_1(k) > RI_2(k) \\ C_2 \text{ if } I_1(k) \leq RI_2(k). \end{cases} \tag{2}$$

For every training pair of data $\mathbf{V}^{in}(k)$, $k = 1, \ldots, K_t$ the error from translating the material response according to rule (2) is

$$\epsilon_{\mathbf{x}}(k) = \begin{cases} 0 \text{ if } & C_M(\mathbf{V}^{in}(k), \mathbf{x}) = C(\mathbf{V}^{in}(k)) \\ 1 \text{ otherwise.} & \end{cases} \tag{3}$$

The mean total error is given by

$$\Phi_e(\mathbf{x}) = \frac{1}{K_t} \sum_{k=1}^{K_t} \epsilon_{\mathbf{x}}(k). \tag{4}$$

Two penalty terms are added to (4), H and U. $H(\mathbf{x})$ penalises solutions with high configuration voltages and is given by

$$H(\mathbf{x}) = \frac{\sum_{j=1}^{8} V_j^2}{8V_{max}^2}. \tag{5}$$

The rationale behind this penalisation is that incremental and generally low levels of configuration voltages are preferable. Solutions where high $V_j^{(k)}$ are applied can destroy material structures favourable to the problem formed during evolution. On the other hand, solutions that render the material unresponsive need to be avoided. A measure of such unresponsiveness is calculated at the end of each search iteration ι, where a sample equal to the population size S of error function evaluations is available. Let $\sigma^2_{o,\iota}$ denote the variance of $\Phi(\mathbf{x})$ and $\sigma^2_{V,\iota}$ the variance of $\sum_{j=1}^{8} V_j^2$ at iteration ι. A value of $\sigma^2_{o,\iota}$ close to zero indicates a non-responsive material and the penalty term takes the form

$$U_\iota = \left(1 - \frac{\sigma^2_{o,\iota}}{\sigma^2_{V,\iota}}\right)^2. \tag{6}$$

Hence, the total objective function $\Phi_s(\mathbf{x})$ for an arbitrary individual s at iteration ι is given by

$$\Phi_s(\mathbf{x}) = \Phi_e(\mathbf{x}) + H(\mathbf{x}) + U_\iota. \tag{7}$$

U_ι aims at leading the optimisation away from material states where the same response is given for different inputs.

The optimisation problem to be solved is that of minimising (7) for a population of size S, subject to voltage bound constraints $V_j \in [V_{\min}, V_{\max}]$, $R > 0$, electrode assignment p and classification rule (2). $V_{\min} = 0$ Volts and for the SC problem $V_{\max} = 4$ Volts whereas for the MC and V1C $V_{\max} = 7$ Volts.

Two different stochatic optimisation algorithms are used for solving this problem, differential evolution (DE) [20] and particle swarm optimisation (PSO) [4]. A constricted version of PSO with parameters taken from [9] is implemented. The DE algorithm implementation uses the parameters suggested in [17]. A population size of $S = 10$ is used for DE and PSO.

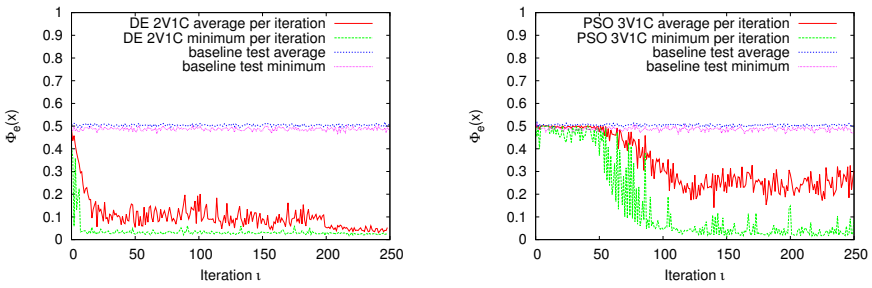
5 Results and Discussion

The first column of Table 1 presents the minimum error Φ_e^* achieved during training. Once training is terminated, verification is performed on the trained material by applying back the optimal solution achieved along with the previously unused verification data. The same verification procedure is repeated ten times. The other four columns of Table 1 refer to results of these runs. $\Phi_{e,v}^*$ is the minimum error, $\Phi_{e,v}^w$ the worst, $\overline{\Phi}_{e,v}$ the average and $\sigma^2_{\Phi_{e,v}}$ the variance. For both DE and PSO, the penalty terms $H(\mathbf{x})$ and U_ι are not included and the classification error for Φ_e is given, for the sake of brevity.

Table 1 shows that for all problems, except outliers, and both algorithms, the observed error increase at the verification phase is $\Phi_e^* - \Phi_{e,v}^* < 2.125\%$. This indicates that the material's behaviour is consistent and generalises well as a classifier. Solutions obtained during training using DE can be better than those of PSO, especially for the SC and V1C datasets. However, PSO outperforms DE with respect to consistency across experiments and generalisation of the solution.

Table 1: Training and verification errors for SC, MC and V1C problems.

SC Experiments	$\Phi_e^*(\%)$	$\Phi_{e,v}^*(\%)$	$\Phi_{e,v}^w(\%)$	$\overline{\Phi}_{e,v}$ (%)	$\sigma^2_{\Phi_{e,v}}$
PSO 1SC	1.3	1.675	2.35	2.0375	0.0527
PSO 2SC	1.6	2.125	3.2	2.6175	0.1277
PSO 3SC	1.3	1.975	2.45	2.25	0.0305
DE 1SC	0.7	1.05	1.625	1.3975	0.03193
DE 2SC	10.4	16.325	18.5	17.4035	0.3652
DE 3SC	1.6	1.675	2.55	2.185	0.06565
MC Experiments	$\Phi_e^*(\%)$	$\Phi_{e,v}^*(\%)$	$\Phi_{e,v}^w(\%)$	$\overline{\Phi}_{e,v}$ (%)	$\sigma^2_{\Phi_{e,v}}$
PSO 1MC	5.8	6.6	7.075	6.815	0.0171
PSO 2MC	5.2	6.325	8.8	7.7225	0.648
PSO 3MC	5.7	7.825	9.025	8.5975	0.1184
DE 1MC	3.4	3.975	4.625	4.38	0.0439
DE 2MC	6.4	7.525	8.95	8.145	0.1739
DE 3MC	5.7	18.25	19.425	18.8375	0.1321
V1C Experiments	$\Phi_e^*(\%)$	$\Phi_{e,v}^*(\%)$	$\Phi_{e,v}^w(\%)$	$\overline{\Phi}_{e,v}$ (%)	$\sigma^2_{\Phi_{e,v}}$
PSO 1V1C	2.7	3.975	5.175	4.6525	0.1318
PSO 2V1C	2.6	3.5	4.25	3.8625	0.0559
PSO 3V1C	1.1	2.525	3.375	2.915	0.063
DE 1V1C	1.3	2.325	2.725	2.4975	0.016
DE 2V1C	1.7	3.125	4.00	3.4975	0.071
DE 3V1C	0.007	4.55	6.2	5.575	0.2617



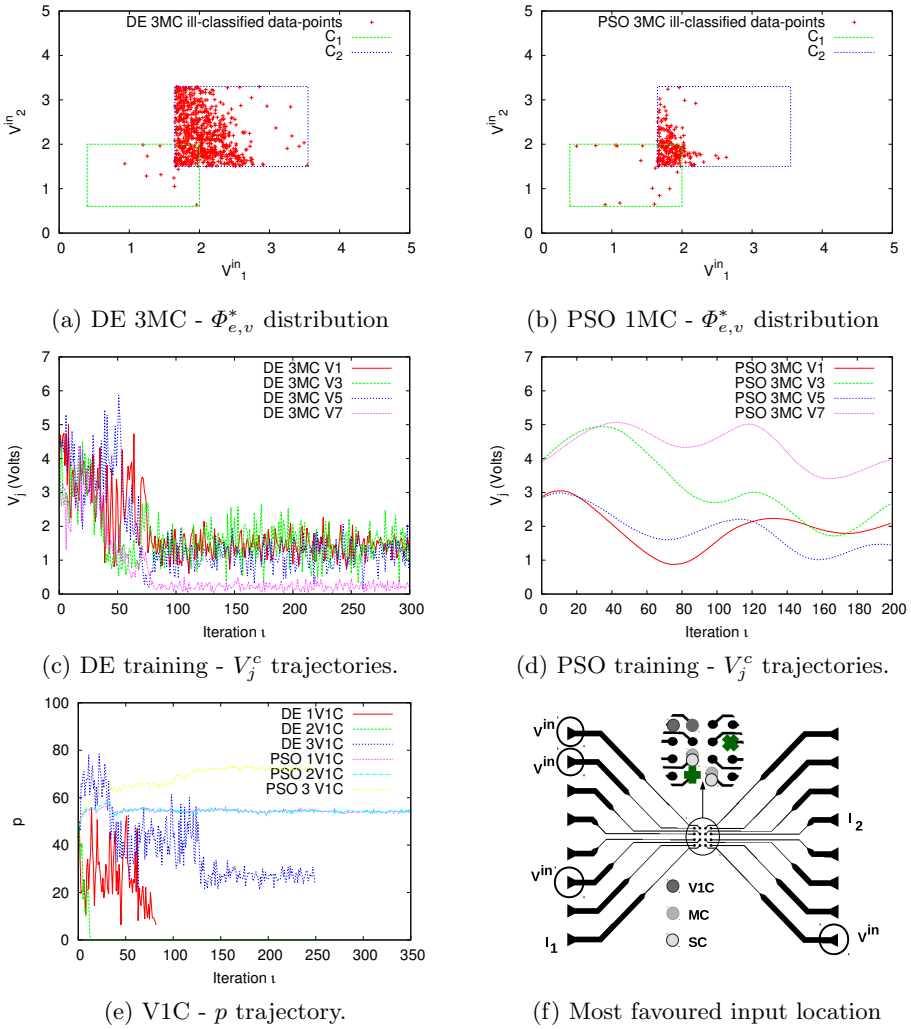
(a) DE 2V1C - $\overline{\Phi}_e(x)$ and Φ_e^* per ι (b) PSO 3V1C - $\overline{\Phi}_e(x)$ and Φ_e^* per ι

Fig. 3: Convergence patterns for training the material based on the V1C data.

This can also be observed for the MC dataset where DE obtains both the smallest and largest error for verification ($\Phi_{e,v}^* = 4.37\%$ and 18.25% respectively) whilst variance of PSO verification tests tends to be lower.

Figure 3 shows the convergence pattern of the error for DE and PSO and is representative of the 18 experiments in Table 1. The baseline tests were performed using samples containing only LCs as material, without any SWCNTs. When DE is used, the material adapts within few iterations; subsequently the algorithm spends more iterations exploiting the minimum found. On the other hand, the PSO algorithm achieves better results at a later stage exploring more the search space.

Figures 4(a) and (b) present the verification error distribution of the 3rd runs of DE and PSO, respectively, using the MC dataset. Both converged to solutions with the same training error $\Phi_e^* = 5.7\%$, but with different $\overline{\Phi}_{e,v}$. The overlapping area of the two classes forms the core of the points that are erroneously classified. However, the better generalisation property of the PSO solution compared to that produced by DE is evident, as the errors outside the overlap are fewer. When



(a) DE 3MC - $\Phi^*_{\epsilon,v}$ distribution

(b) PSO 1MC - $\Phi^*_{\epsilon,v}$ distribution

(c) DE training - V_j^c trajectories.

(d) PSO training - V_j^c trajectories.

(e) V1C - p trajectory.

(f) Most favoured input location

Fig. 4: Visualisation of p and sample V_j^c trajectories for PSO and DE.

the DE solution is applied, the errors are far more widely dispersed and densely distributed into the area of C_2, making a poor classifier out of the material.

Figures 4 (c) and (d) show the distinctive difference between the two algorithm's configuration voltages' trajectories, averaged over S, per iteration. It can be seen that the search performed by DE is more noisy even when the algorithm aims to exploit a minimum. On the other hand, PSO's exploration of the search space is based on smoother inputs. Figure 4 (e) depicts the convergence trajectory of p for all experiments using the V1C dataset. Convergence is not towards the same value of p, but resulting input pin location is similar. Figure

4 (f) presents the corresponding mapping of p with regard to input location on the micro-electrode array for the optimal solutions of the three problems. Experiments resulting to errors between 4-10% for Φ_e^* and $\Phi_{e,v}^*$ tend to have a p corresponding to the most favoured locations shown in Figure 4(f).

Our current hypothesis is that the poorer generalisation of the solutions obtained by DE is due to the pattern of average configuration voltages per iteration. The PSO algorithms smoother trajectories of V_j^c build structures inside the material, reinforcing responses minimising the classification error. The noisy V_j^c applied by DE appears to make the formation of such structures more difficult. Over the different experiments, DE is less consistent in its performance; exploration of the search space by the PSO algorithm results in better conductive circuit formation within the material. This hypothesis needs to be supported by more experiments and evidence, such as image analysis of the material before and after training.

6 Conclusion

This paper has presented the results of an investigation on evolution in materio for a mixture of single walled carbon nanotubes and liquid crystals. Under the influence of different levels of voltage applied at various locations of its body, conductive networks are formed by the nanotubes. Three simple classification problems are considered and training of the material as a data classifier is formulated as an optimisation problem. Results obtained with training and verification datasets are reported, showing that the solution can perform classification for similar problems with different instances. The stronger exploration element of PSO and the smoother input signals sent appear to result to classifiers that generalise better.

This is quite a new area of research and many issues need to be addressed. A more detailed investigation needs to be performed on the optimisation algorithms used and the impact of their search pattern on the solutions' quality. More recent variants of evolution-inspired algorithms need to be implemented as well. The impact of the SWCNT and LC concentration in the mix needs to be evaluated. Finally, more complicated problems will be considered and it would be very interesting to observe the material structure patterns formed for this purpose in each particular case.

References

1. M. Amos, D. Hodgson, and A. Gibbons. Bacterial self-oranisation and computation. *International Journal of Unconventional Computing*, 3(3):199–210, 2007.
2. J. Bird and E. Di Paolo. Gordon pask and his maverick machines. In P. Husbands, O. Holland, and M. Wheeler, editors, *The mechanical mind in history*, chapter 8, pages 185–211. The MIT Press, 2008.
3. S. Bose, C. Lawrence, Z. Liu, K. Makarenko, R. van Damme, H. Broersma, and W. van der Wiel. Evolution of a designless nanoparticle network into reconfigurable boolean logic. *Nature Nanotechnology*, 2015.

4. R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the 6th International Symposium on Micro-machine and Human Science*, volume 1, pages 39–43. New York, NY, 1995.
5. A. Goudarzi, M. R. Lakin, and D. Stefanovic. Reservoir computing approach to robust computation using unreliable nanoscale networks. In *Unconventional Computation and Natural Computation*, pages 164–176. Springer, 2014.
6. S. L. Harding and J. F. Miller. Evolution in materio: Computing with liquid crystal. *International Journal of Unconventional Computing*, 3(4):243–257, 2007.
7. J. Jones, J. G. Whiting, and A. Adamatzky. Quantitative transformation for implementation of adder circuits in physical systems. *Biosystems*, 134:16–23, 2015.
8. A. Kotsialos, M. K. Massey, F. Qaiser, D. Zeze, C. Pearson, and M. C. Petty. Logic gate and circuit training on randomly dispersed carbon nanotubes. *International Journal of Unconventional Computing*, 10(5-6):473–497, 2014.
9. E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis. Particle swarm optimization for integer programming. In *WCCI*, pages 1582–1587. IEEE, 2002.
10. M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.
11. M. K. Massey, A. Kotsialos, F. Qaiser, D. A. Zeze, C. Pearson, D. Volpati, L. Bowen, and M. C. Petty. Computing with carbon nanotubes: Optimization of threshold logic gates using disordered nanotube/polymer composites. *Journal of Applied Physics*, 117(13):134903, 2015.
12. J. F. Miller and K. Downing. Evolution in materio: Looking beyond the silicon box. In *Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware*, pages 167–176. IEEE, 2002.
13. J. F. Miller, S. L. Harding, and G. Tufte. Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence*, 7(1):49–67, 2014.
14. J. F. Miller and M. Mohid. Function optimization using cartesian genetic programming. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 147–148. ACM, 2013.
15. M. Mohid, J. F. Miller, S. L. Harding, G. Tufte, O. R. Lykkebø, M. K. Massey, and M. C. Petty. Evolution-in-materio: Solving machine learning classification problems using materials. In *Parallel Problem Solving from Nature–PPSN XIII*, pages 721–730. Springer, 2014.
16. NASCENCE project (ICT 317662). Report on suitable computational tasks of various difficulties, 2013. Deliverable D4.2.
17. M. E. H. Pedersen. Good parameters for differential evolution. Technical report, Technical report, Hvass Computer Science Laboratories, 2010.
18. S. Prasad, M. Yang, X. Zhang, C. S. Ozkan, and M. Ozkan. Electric field assisted patterning of neuronal networks for the study of brain functions. *Biomedical Microdevices*, 5(2):125–137, 2003.
19. S. Stepney. The neglected pillar of material computation. *Physica D: Nonlinear Phenomena*, 237(9):1157–1164, 2008.
20. R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.
21. A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In *Evolvable systems: from biology to hardware*, pages 390–405. Springer, 1996.
22. D. Volpati, M. K. Massey, D. Johnson, A. Kotsialos, F. Qaiser, C. Pearson, K. Coleman, G. Tiburzi, D. A. Zeze, and M. C. Petty. Exploring the alignment of carbon nanotubes dispersed in a liquid crystal matrix using coplanar electrodes. *Journal of Applied Physics*, 117(12):125303, 2015.